Geometric Modeling

Cotangent Laplacian Derivation

Julian Panetta — fjp234@nyu.edu

NYU | COURANT

Administrative Details

- **HW2 demo session moved to next week!**
  (Next Friday is the deadline for midterm grades)

- **New date: March 29, 2-3PM**

- **Live demo is REQUIRED to get a grade for HW2**

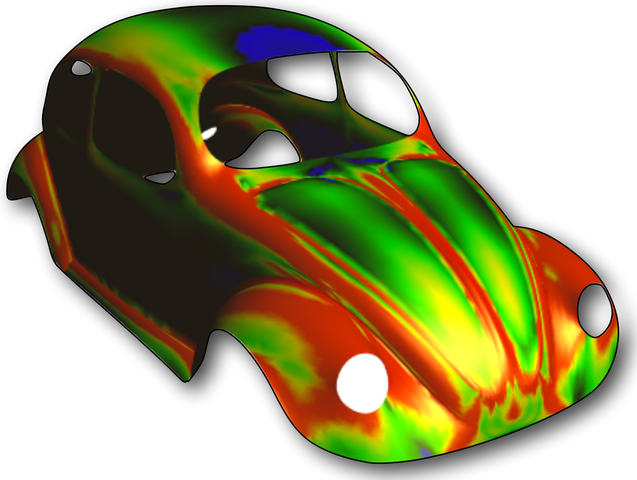
  - Email me to schedule another time if you cannot make it: fjp234@nyu.edu

- HW4 out soon.

Cotan Laplacian Motivation

- Discrete Laplacian enables many useful operations on triangle meshes:

$$H_i = \frac{1}{2}\left\| [\mathbf{Lv}]_i \right\|$$

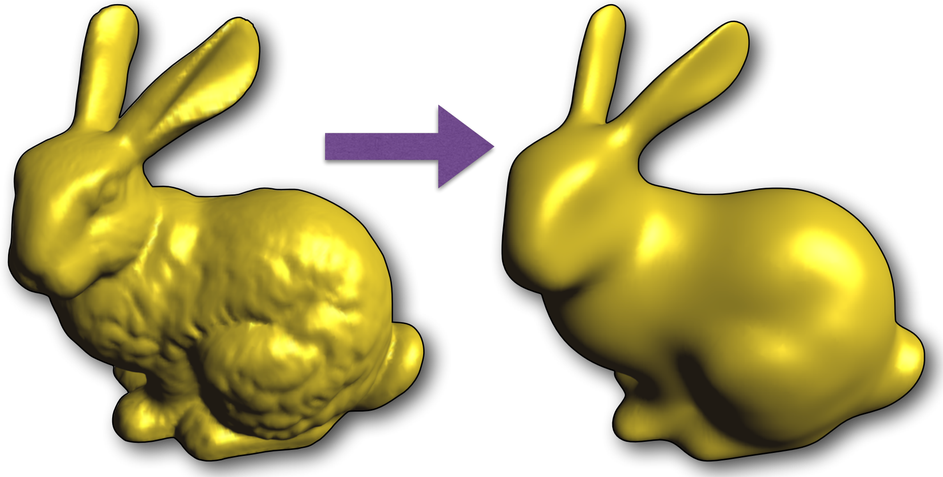


Mean Curvature

$$\mathbf{n} = -\frac{[\mathbf{Lv}]_i}{2H_i}$$

Normal Computation

$$(I - \lambda dt\mathbf{L})\mathbf{v}^{n+1} = \mathbf{v}^n$$



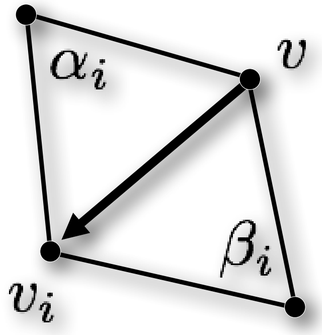
Smoothing

...

- Cotangent weights give **less mesh-dependent** results, ensuring **proper convergence** under mesh refinement

Where We're Headed

$$\Delta_M^{\text{cotan}} f(\mathbf{v}) = \frac{1}{2A(\mathbf{v})} \sum_{\mathbf{v}_i \in N(\mathbf{v})} (\cot(\alpha_i) + \cot(\beta_i)) \Big(f(\mathbf{v}_i) - f(\mathbf{v}) \Big)$$



- "Cotan Laplacian" refers to $\frac{1}{2}(\cot(\alpha_i) + \cot(\beta_i))$ weight on each edge contribution.
- We'll consider two different ways to derive this formula:
  - "Traditional graphics approach:" Finite Volume-style
  - Finite Element Approach (more general)

Goal: Mesh Independence

- We want discrete Laplacian to **converge to the Laplace Beltrami operator** of smooth surface S as mesh M is refined to better approximate S.

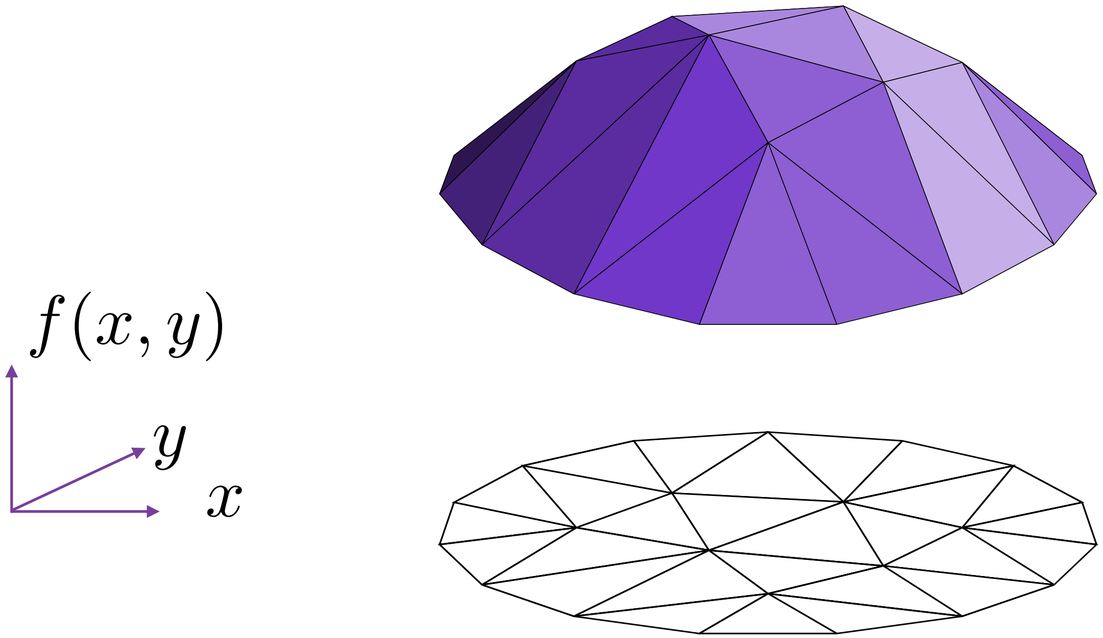
$$\Delta_M^{\text{cotan}}\mathbf{F} \quad \longrightarrow \quad \Delta_S f \qquad\qquad \mathbf{F} = \begin{bmatrix} f(\mathbf{v}_1) \\ f(\mathbf{v}_2) \\ f(\mathbf{v}_3) \\ \cdots \\ f(\mathbf{v}_N) \end{bmatrix}$$

- **F** is a vector of discrete values (per vertex), f(x) is a smooth function.

- **Idea:** reinterpret **F** as a **function interpolating the smooth** “f,” and directly compute its Laplacian. Converge as **F** —> f

Piecewise-Linear Interpolation

- Since **F** holds per-vertex values, it naturally corresponds to a **linear function on each triangle**



- **(Corner values uniquely determine linear function over triangle)**

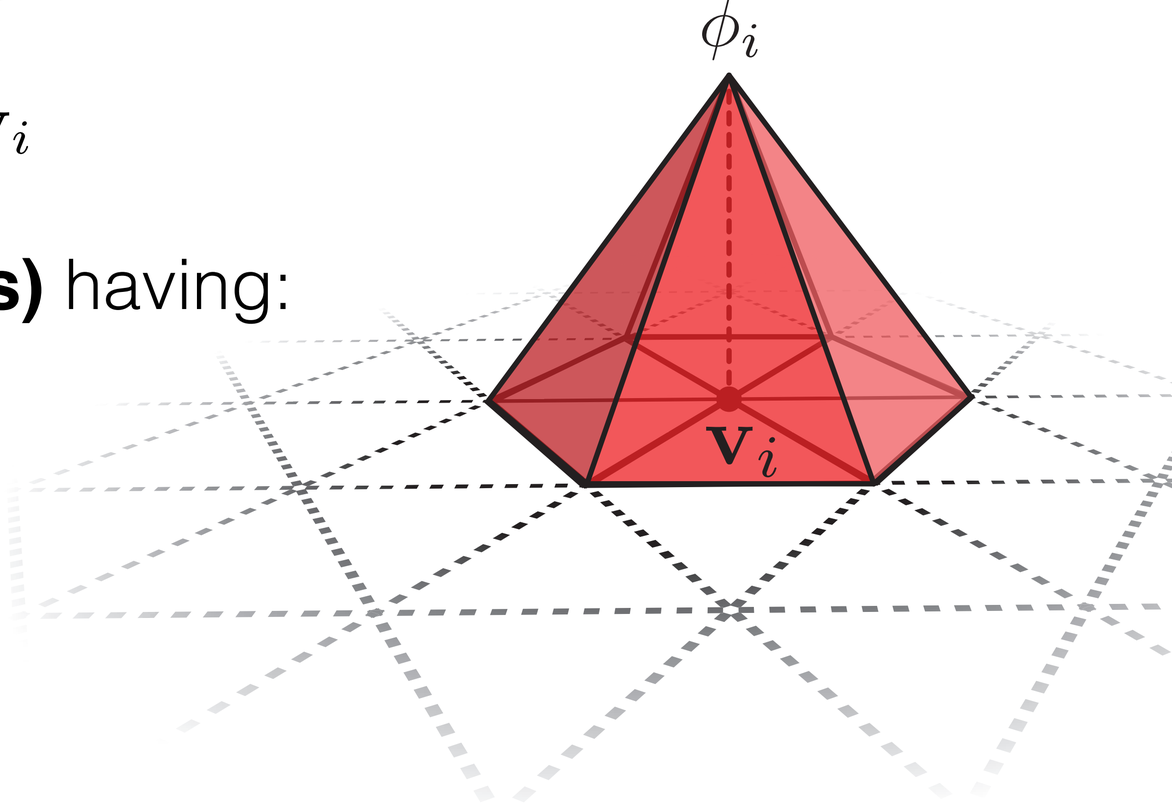
“Hat Function” Basis

- Piecewise linear functions are expressed most simply using the “**hat functions**” (linear Lagrange polynomials)
- One basis function $\phi_i(\mathbf{x})$ per vertex, $\mathbf{v}_i$
- **Unique linear function (on triangles)** having:

$$\phi_i(\mathbf{v}_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

- Interpolation of **F** on mesh M is just:

$$f_M(\mathbf{x}) = \sum_i f(\mathbf{v}_i)\phi_i(\mathbf{x})$$



Gradients

- Recall, the Laplacian is the divergence of the gradient:

$$\Delta f = \nabla \cdot (\nabla f)$$

- So we first need to compute our interpolant's **gradient:**

$$\nabla f_M(\mathbf{x}) = \nabla \left(\sum_i f(\mathbf{v}_i)\phi_i(\mathbf{x}) \right) = \sum_i f(\mathbf{v}_i)\nabla\phi_i(\mathbf{x})$$

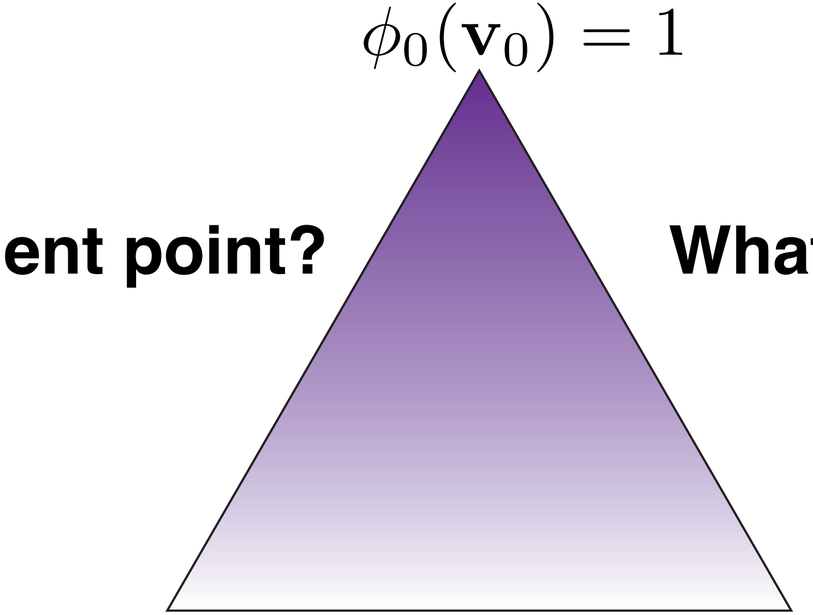
- In other words, the gradient is just a linear combination of the **basis function gradients** $\nabla\phi_i(\mathbf{x})$ weighted by the per-vertex f values.

Hat Function Gradients

- Basis functions are linear over each triangle, so these gradients are **constant on each triangle**

- We can determine gradient with simple geometric arguments:

Where does the gradient point?



What about its magnitude?

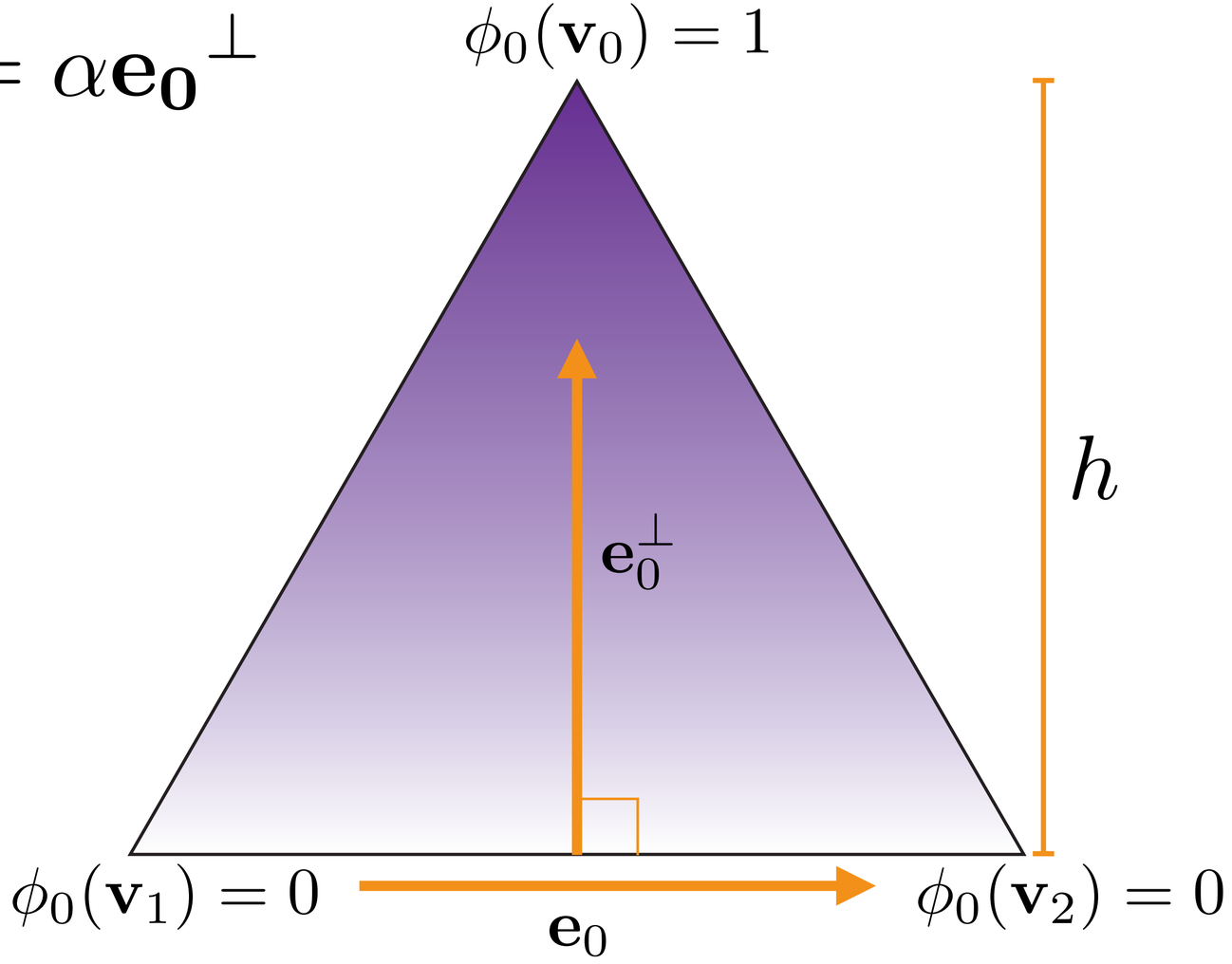
Gradient Derivation

$$\nabla\phi_0 \cdot \mathbf{e}_0 = 0 \quad \Longrightarrow \quad \nabla\phi_0 = \alpha \mathbf{e_0}^{\perp}$$

$$\nabla\phi_0 \cdot \left(h \frac{\mathbf{e}_0^{\perp}}{\|\mathbf{e}_0^{\perp}\|} \right) = 1$$

$$\Longrightarrow \quad \nabla\phi_0 = \frac{1}{h} \frac{\mathbf{e}_0^{\perp}}{\|\mathbf{e}_0^{\perp}\|}$$

$$\Longrightarrow \quad \nabla\phi_0 = \frac{\mathbf{e}_0^{\perp}}{2A}$$



(area is 1/2 * base * height)

Final Gradient Expression

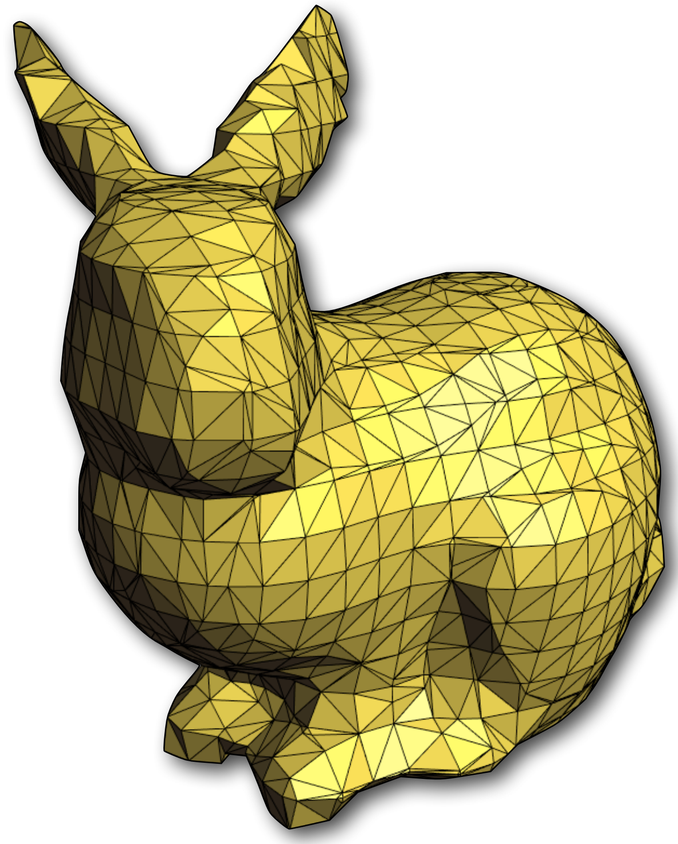
$$\nabla \phi_i|_T = \frac{\mathbf{e}_i^\perp}{2A_T}$$

- Note, this is the gradient **evaluated on one triangle T incident vertex i.** (Different gradient on others).

- Works on curved triangle meshes too! Just need triangle area and inward-pointing perpendicular edge vectors: $\mathbf{e}_i^\perp = \mathbf{n} \times \mathbf{e}_i$

- Full gradient is then the linear combination

$$\nabla f_M(\mathbf{x}) = \sum_i f(\mathbf{v}_i) \nabla \phi_i(\mathbf{x})$$

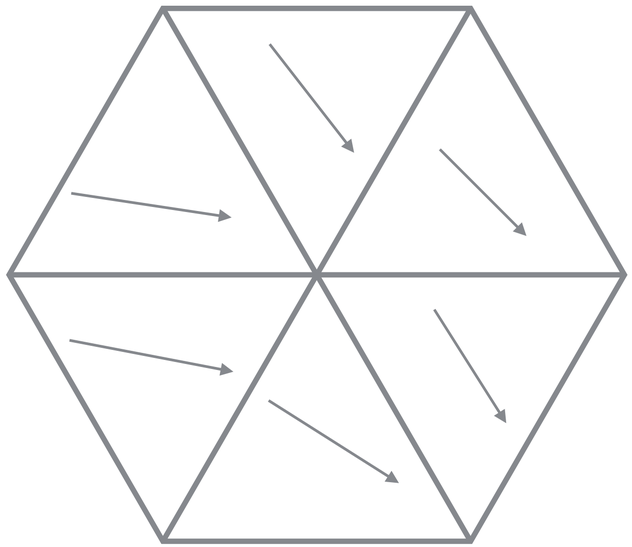


Now the Laplacian!

- Now we want to compute

$$\Delta f_M = \nabla \cdot \nabla f_M(\mathbf{x}) = \nabla \cdot \left(\sum_i f(\mathbf{v}_i) \nabla \phi_i(\mathbf{x}) \right)$$

- But the gradient is piecewise constant!
  What does divergence even mean?

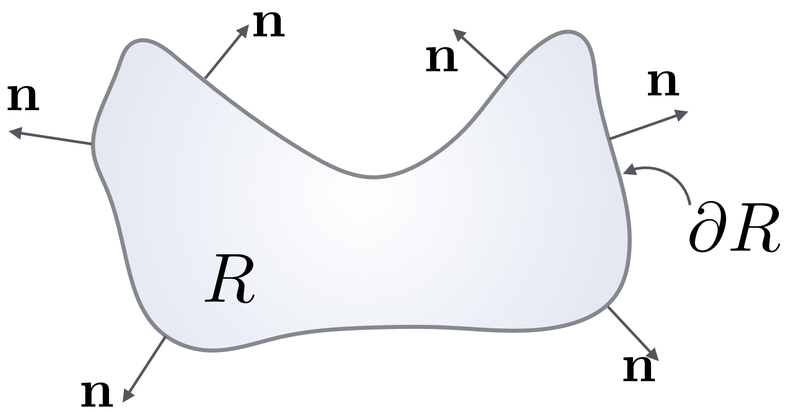


Too Many Derivatives!

- **Problem:** function is piecewise linear (only C0 continuous) and we're trying to take second derivatives

- But we can make sense of things by using the Divergence Theorem over some region "R":

$$\int_R \nabla \cdot (\nabla \phi)\, \mathrm{d}\mathbf{x} = \int_{\partial R} \mathbf{n} \cdot \nabla \phi \, \mathrm{d}s$$



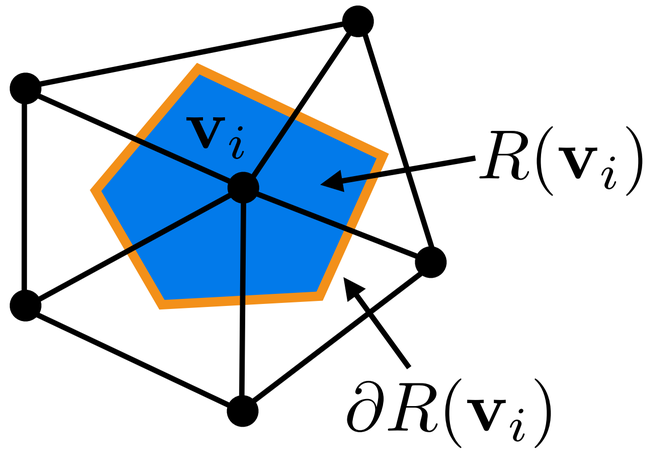
- (This allows us to avoid explicitly computing divergence; just need dot products with the curve normals on R's boundary.)

Approach 1: (Traditional in Graphics)

- Idea: define Laplacian on a vertex to be the **average over some region surrounding the vertex**.

$$[L\mathbf{F}]_i := \frac{1}{|R(\mathbf{v}_i)|} \int_{R(\mathbf{v}_i)} \nabla \cdot \nabla f_M(\mathbf{x}) \, \mathrm{d}\mathbf{x}$$

$$= \frac{1}{|R(\mathbf{v}_i)|} \int_{\partial R(\mathbf{v}_i)} \mathbf{n} \cdot \nabla f_M(\mathbf{x}) \, \mathrm{d}s$$



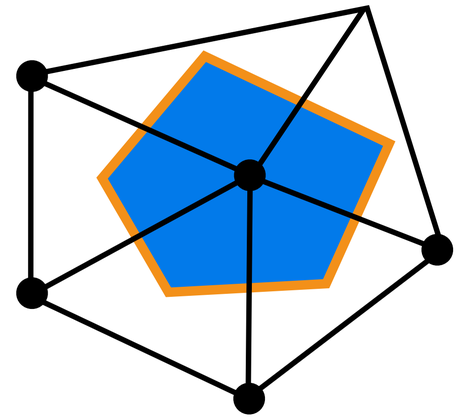
- To get cotan Laplacian, you need to average over the **vertices' Voronoi regions** (region for which v is the closest vtx).

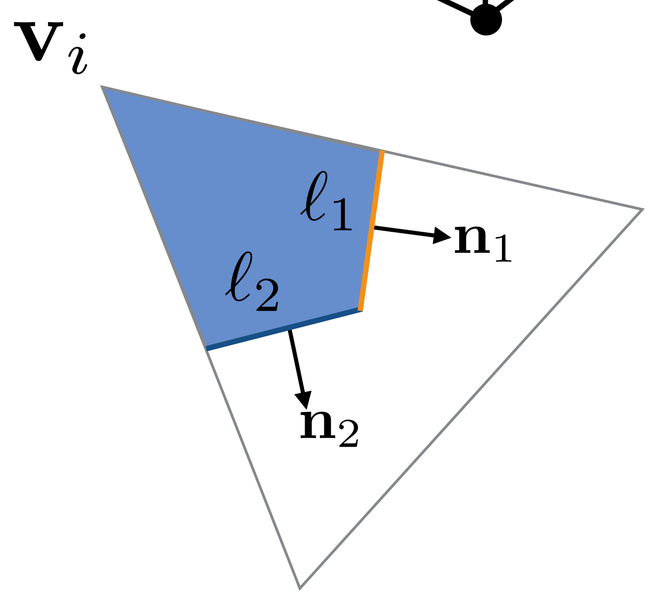
- This region seems somewhat arbitrary; approach 2 won't need this choice.

Approach 1 Continued

- Break into sum over triangles (so gradient is constant):

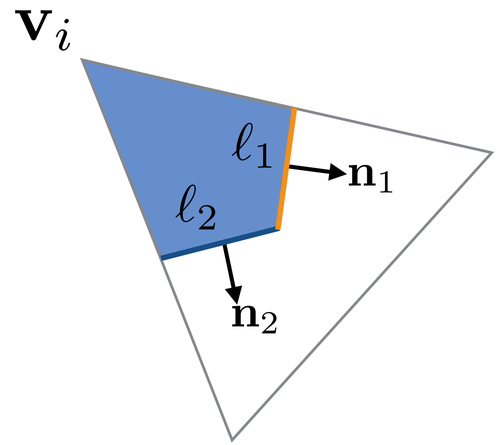
$$
\begin{aligned}
[L\mathbf{F}]_i &= \frac{1}{|R(\mathbf{v}_i)|} \int_{\partial R(\mathbf{v}_i)} \mathbf{n} \cdot \nabla f_M(\mathbf{x}) \, \mathrm{d}s \\
&= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \int_{T \cap \partial R(\mathbf{v}_i)} \mathbf{n} \cdot \nabla f_M(\mathbf{x}) \, \mathrm{d}s \\
&= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \nabla f_T \cdot \int_{T \cap \partial R(\mathbf{v}_i)} \mathbf{n} \, \mathrm{d}s \\
&= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \nabla f_T \cdot (\mathbf{n}_1 \ell_1 + \mathbf{n}_2 \ell_2)
\end{aligned}
$$



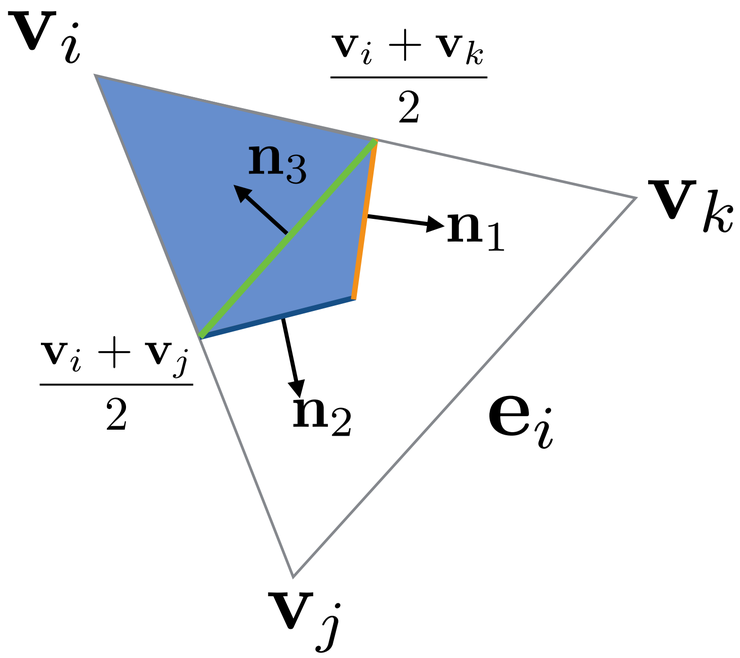


Voronoi Diagram Properties

$$[L\mathbf{F}]_i = \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \nabla f_T \cdot (\mathbf{n}_1 \ell_1 + \mathbf{n}_2 \ell_2)$$



- Voronoi diagram edges **bisect** v_i's incident edges



$$\mathbf{n}_1 \ell_1 + \mathbf{n}_2 \ell_2 + \mathbf{n}_3 \ell_3 = 0$$

(Tri edge vectors sum to zero; so do their perpendiculars.)

$$\implies \quad \mathbf{n}_1 \ell_1 + \mathbf{n}_2 \ell_2 = -\mathbf{n}_3 \ell_3 = \left(\frac{\mathbf{v}_k + \mathbf{v}_i}{2} - \frac{\mathbf{v}_j + \mathbf{v}_i}{2} \right)^{\perp}$$

$$= \frac{1}{2} (\mathbf{v}_k - \mathbf{v}_j)^{\perp} = \frac{\mathbf{e}_i^{\perp}}{2}$$

$$\implies \quad [L\mathbf{F}]_i = \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \nabla f_T \cdot \frac{\mathbf{e}_i^{\perp}}{2}$$

Plug in per-Tri Gradient

$$[L\mathbf{F}]_i = \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \nabla f_T \cdot \frac{\mathbf{e}_i^\perp}{2}$$

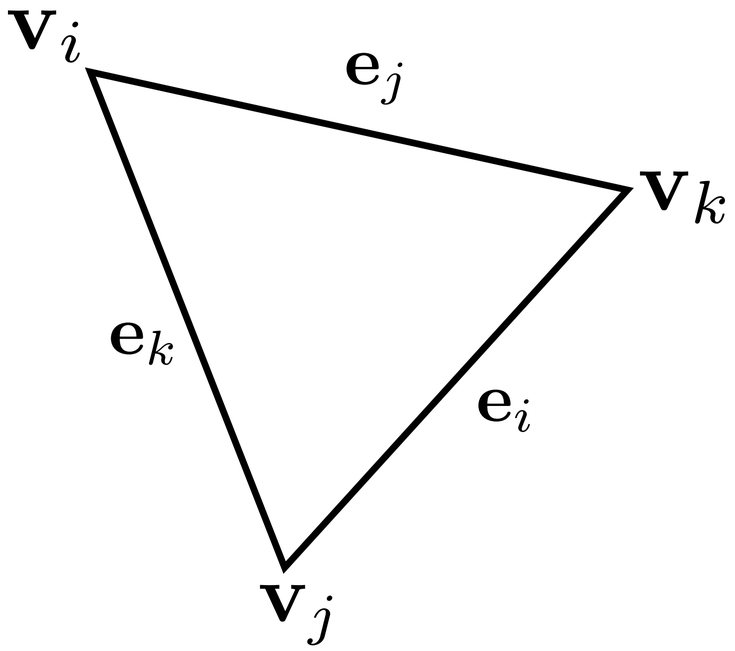
$$= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} (f_i \nabla \phi_i + f_j \nabla \phi_j + f_k \nabla \phi_k) \cdot \frac{\mathbf{e}_i^\perp}{2}$$

$$= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \left[f_i \frac{\mathbf{e}_i^\perp}{2A_T} + f_j \frac{\mathbf{e}_j^\perp}{2A_T} + f_k \frac{\mathbf{e}_k^\perp}{2A_T} \right] \cdot \frac{\mathbf{e}_i^\perp}{2}$$

$$= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \left[-f_i \left(\frac{\mathbf{e}_j^\perp}{2A_T} + \frac{\mathbf{e}_k^\perp}{2A_T} \right) + f_j \frac{\mathbf{e}_j^\perp}{2A_T} + f_k \frac{\mathbf{e}_k^\perp}{2A_T} \right] \cdot \frac{\mathbf{e}_i^\perp}{2}$$

(edges sum to zero: $\mathbf{e}_i^\perp + \mathbf{e}_j^\perp + \mathbf{e}_k^\perp = 0$)

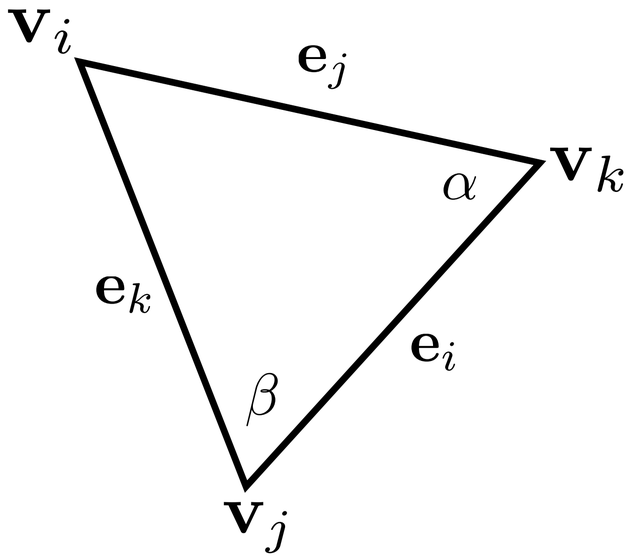
$$= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \left[(f_j - f_i) \frac{\mathbf{e}_j^\perp}{2A_T} + (f_k - f_i) \frac{\mathbf{e}_k^\perp}{2A_T} \right] \cdot \frac{\mathbf{e}_i^\perp}{2}$$



Final Steps

$$[L\mathbf{F}]_i = \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \left[(f_j - f_i) \frac{\mathbf{e}_j^\perp}{2A_T} + (f_k - f_i) \frac{\mathbf{e}_k^\perp}{2A_T} \right] \cdot \frac{\mathbf{e}_i^\perp}{2}$$

$$= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} (f_j - f_i) \frac{\mathbf{e}_j^\perp \cdot \mathbf{e}_i^\perp}{4A_T} + (f_k - f_i) \frac{\mathbf{e}_k^\perp \cdot \mathbf{e}_i^\perp}{4A_T}$$



Note:

$$\frac{\mathbf{e}_i^\perp \cdot \mathbf{e}_j^\perp}{4A_T} = \frac{\|\mathbf{e}_i\| \|\mathbf{e}_j\| \cos(\alpha)}{2\|\mathbf{e}_i\| \|\mathbf{e}_j\| \sin(\alpha)} = \frac{1}{2} \cot(\alpha)$$

$$\frac{\mathbf{e}_i^\perp \cdot \mathbf{e}_k^\perp}{4A_T} = \frac{\|\mathbf{e}_i\| \|\mathbf{e}_k\| \cos(\beta)}{2\|\mathbf{e}_i\| \|\mathbf{e}_k\| \sin(\beta)} = \frac{1}{2} \cot(\beta)$$

$$[L\mathbf{F}]_i = \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} (f_j - f_i) \cot(\alpha) + (f_k - f_i) \cot(\beta)$$

Regrouping Terms

- We have the cotan weights now, but expressed as a sum over triangles:

$$[L\mathbf{F}]_i = \frac{1}{2|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} (f_j - f_i)\cot\alpha + (f_k - f_i)\cot\beta$$

- To get the standard formula, we can regroup these terms as a sum over edges:

$$[L\mathbf{F}]_i = \frac{1}{2|R(\mathbf{v}_i)|} \sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} (\cot(\alpha_{ij}) + \cot(\beta_{ij}))\Big(f(\mathbf{v}_j) - f(\mathbf{v}_i)\Big)$$

- $|R(v_i)|$ is the area of the i^{th} vertex's Voronoi region (the averaging area)

Approach 2: Finite Element Method

- Instead of trying to discretize the Laplacian itself (using averaging regions), we **discretize the whole PDE** containing it. For instance:

$$-\Delta u = b \quad \text{in } M$$

- Here “b” is a forcing term (e.g. injecting/sinking heat at each point), and we assume M is a closed surface mesh without a boundary.

- If M had boundary curves, we’d need to specify boundary conditions.

Weak Form of the Poisson Equation

- We can't expect a piecewise linear function "u" to solve this PDE exactly (why does its Laplacian mean, anyway?)
- But we can ask it to be a "weak solution:"

$$-\int_M \psi \Delta u \, \mathrm{d}\mathbf{x} = \int_M \psi b \, \mathrm{d}\mathbf{x} \qquad \forall \psi \text{ "test functions"}$$

- In other words, we ask for zero residual "along" all test functions.
- Linear FEM: choose piecewise linear functions for both u and ψ
- Physical interpretation: minimizing a potential energy over the space of piecewise linear functions

Expand Functions in Basis

$$-\int_M \psi \Delta u \, \mathrm{d}\mathbf{x} = \int_M \psi b \, \mathrm{d}\mathbf{x}$$

$$u(\mathbf{x}) := \sum_j u_j \phi_j(\mathbf{x}) \qquad \psi(\mathbf{x}) := \sum_i a_i \phi_i(\mathbf{x}) \qquad b(\mathbf{x}) := \sum_j b_j \phi_j(\mathbf{x})$$

$$-\int_M \left(\sum_i a_i \phi_i(\mathbf{x}) \right) \Delta \left(\sum_j u_j \phi_j(\mathbf{x}) \right) \, \mathrm{d}\mathbf{x} = \int_M \left(\sum_i a_i \phi_i(\mathbf{x}) \right) \left(\sum_j b_j \phi_j(\mathbf{x}) \right) \, \mathrm{d}\mathbf{x}$$

$$\implies \quad -\sum_{i,j} a_i u_j \left(\int_M \phi_i \Delta \phi_j \, \mathrm{d}\mathbf{x} \right) = \sum_{i,j} a_i b_j \left(\int_M \phi_i \phi_j \, \mathrm{d}\mathbf{x} \right)$$

Integration by Parts for $\int_M \phi_i \Delta \phi_j \, \mathrm{d}\mathbf{x}$

- To avoid second derivatives, apply **divergence theorem** in the form of an **integration by parts** (move derivative onto ϕ_i)

- Note the "product rule": $\nabla \cdot (\phi_i \nabla \phi_j) = \phi_i \Delta \phi_j + \nabla \phi_i \cdot \nabla \phi_j \quad \Longrightarrow$

$$\phi_i \Delta \phi_j = \nabla \cdot (\phi_i \nabla \phi_j) - \nabla \phi_i \cdot \nabla \phi_j$$

Apply divergence theorem

$$\begin{aligned} -\int_M \phi_i \Delta \phi_j \, \mathrm{d}\mathbf{x} &= -\int_M \nabla \cdot (\phi_i \nabla \phi_j) - \nabla \phi_i \cdot \nabla \phi_j \, \mathrm{d}\mathbf{x} \\ &= -\int_{\partial M} \mathbf{n} \cdot (\phi_i \nabla \phi_j) \, \mathrm{d}s + \int_M \nabla \phi_i \cdot \nabla \phi_j \, \mathrm{d}\mathbf{x} \\ &= \int_M \nabla \phi_i \cdot \nabla \phi_j \, \mathrm{d}\mathbf{x} \end{aligned}$$

There is no boundary!

Laplacian and Mass Matrices

- Now our weak PDE looks like:

$$\sum_{i,j} a_i u_j \underbrace{\left(\int_M \nabla \phi_i \nabla \phi_j \, \mathrm{d}\mathbf{x} \right)}_{L_{ij}} = \sum_{i,j} a_i b_j \underbrace{\left(\int_M \phi_i \phi_j \, \mathrm{d}\mathbf{x} \right)}_{\mathcal{M}_{ij}}$$

- Expressed in matrix form:

$$\mathbf{a}^T L \mathbf{u} = \mathbf{a}^T \mathcal{M} \mathbf{b} \quad \boxed{\forall \mathbf{a}}$$

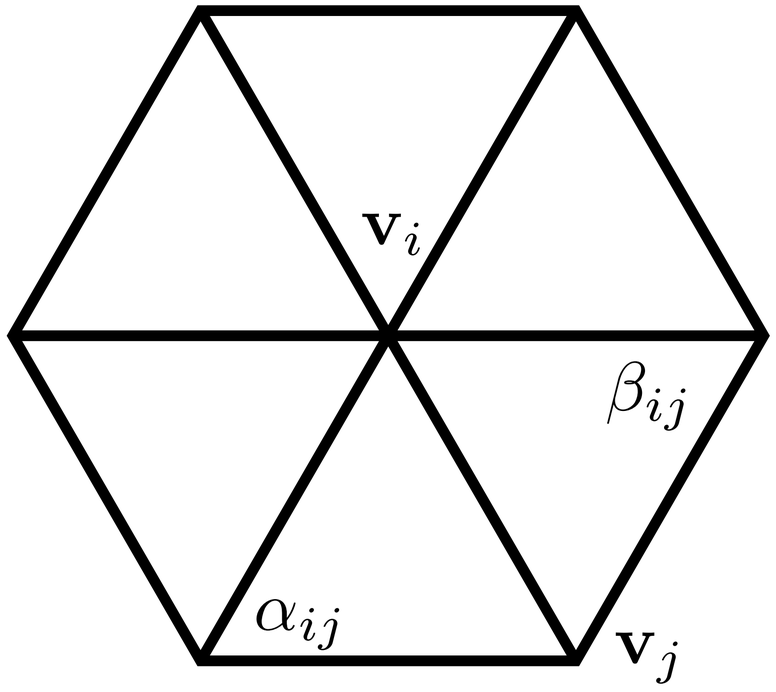
Weak form must hold for all test functions!

$$\implies \quad L\mathbf{u} = \mathcal{M}\mathbf{b}$$

- Plugging in gradient expressions:

$$L_{ij} = \int_M \nabla \phi_i \cdot \nabla \phi_j \, \mathrm{d}\mathbf{x} = \sum_{T \ni \mathbf{v}_i, \mathbf{v}_j} \frac{\mathbf{e}_{T,i}^{\perp}}{2A_T} \cdot \frac{\mathbf{e}_{T,j}^{\perp}}{2A_T} A_T = \sum_{T \ni \mathbf{v}_i, \mathbf{v}_j} \frac{\mathbf{e}_{T,i}^{\perp} \cdot \mathbf{e}_{T,j}^{\perp}}{4A_T}$$

Laplacian Matrix Cotangent Weights



$$L_{ij} = \sum_{T \ni \mathbf{v}_i, \mathbf{v}_j} \frac{\mathbf{e}^{\perp}_{T,i} \cdot \mathbf{e}^{\perp}_{T,j}}{4A_T}$$

$$L_{ij} = \begin{cases} \frac{1}{2}\left(\cot(\alpha_{ij}) + \cot(\beta_{ij})\right) & \text{if } i \neq j \\ \sum_{T \ni \mathbf{v}_i} \frac{\mathbf{e}^{\perp}_{T,i} \cdot \left(-\mathbf{e}^{\perp}_{T,j} - \mathbf{e}^{\perp}_{T,k}\right)}{4A_T} = -\frac{1}{2} \sum_{\mathbf{v}_k \in N(\mathbf{v}_i)} \left(\cot(\alpha_{ik}) + \cot(\beta_{ik})\right) & \text{if } i = j \end{cases}$$

FEM vs “Graphics Approach”

- Get exactly the same cotan weights (up to division by region area)
- FEM doesn’t require defining an averaging region
- FEM computes Laplacian **integrated against test function** instead of an **averaged** (point) quantity:

$$L_{\mathrm{FEM}} = M_{\mathrm{diag}} L_{\mathrm{Avg}} \qquad M_{\mathrm{diag}} = \mathrm{diag}([R(\mathbf{v}_0), R(\mathbf{v}_1), \cdots])$$

- FEM uses **exact mass matrix** to integrate RHS against test functions; graphics approach uses **diagonal “Lumped” approximation** M_{diag} based on region areas (e.g. Voronoi)—still converges, but higher error.
  (Lumping effectively treats the RHS function “b” as **piecewise constant** over the averaging region, while FEM+exact mass matrix treats it as **piecewise linear**).
- FEM generalizes to higher degree; just use more basis functions!